Extraction de motifs temporels à partir de séquences d'événements avec intervalles temporels

Thomas Guyet*, René Quiniou**

*AGROCAMPUS-OUEST
65 rue de Saint-Brieuc, CS 84215, 35 042 Rennes Cedex
thomas.guyet@agrocampus-ouest.fr,
**INRIA, Centre de Rennes - Bretagne Atlantique
Campus de Beaulieu, F - 35 042 Rennes Cedex
rene.quiniou@irisa.fr

Résumé. La fouille de base de données séquentielles a pour objet l'extraction de motifs séquentiels représentatifs. La plupart des méthodes concernent des motifs composés d'événements liés par des relations temporelles basées sur la précédence des instants. Pourtant, dans de nombreuses situations réelles une information quantitative sur la durée des événements ou le délai inter-événements est nécessaire pour discriminer les phénomènes. Nous proposons deux algorithmes, QTIAPriori et QTIPrefixSpan, pour extraire des motifs temporels composés d'événements associés à des intervalles décrivant leur position dans le temps et leur durée. Chacun d'eux ajoute aux algorithmes GSP et PrefixSpan une étape de catégorisation d'intervalles multi-dimensionnels pour extraire les intervalles temporelles représentatifs. Les expérimentations sur des données simulées montrent la capacité des algorithmes à extraire des motifs précis en présence de bruit et montrent l'amélioration des performances en temps de calcul.

1 Introduction

Dans de nombreux domaines d'application, médecine (Quiniou et al. (2003)), agronomie (Le Ber et al. (2006)), e-commerce (Masseglia et al. (2009)), industrie (Giannotti et al. (2006)), IHM (Cram et al. (2009)), les méthodes de fouille de données sont utilisées pour extraire des motifs séquentiels à partir de séquences. La fouille de séquences s'est principalement intéressée à l'extraction de motifs dont la dimension temporelle ou spatiale repose sur un ordonnancement simple des items (occurrences dans le temps ou dans l'espace des objets composant les motifs). Cependant, dans de nombreux domaines d'application un tel ordonnancement n'est pas suffisant : une caractérisation quantitative des durées des événements ou des délais entre les occurrences de ces événements est nécessaire pour affiner les motifs extraits et produire les connaissances utilisables pour différencier certains comportements. Par exemple, deux achats effectués à un jour, un mois ou un an d'intervalle ne sont pas corrélés de la même façon.

Notre objectif est de définir une méthode de fouille de séquences temporelles composées d'occurrences d'événements pouvant avoir une durée. Le but est d'extraire les motifs séquen-

tiels intéressants, ici fréquents, composés d'événements auxquels sont associés des intervalles temporels numériques caractérisant à la fois leur durée et leur position relative dans le temps.

Dans cet article, nous proposons deux algorithmes efficaces pour l'extraction de séquences d'intervalles temporels utilisant successivement une méthode de fouille de séquences et une caractérisation des intervalles à partir d'un algorithme de catégorisation d'intervalles. Nous comparons plusieurs méthodes et distances issues de l'analyse de données symboliques pour extraire les caractéristiques temporelles quantitatives représentatives des séquences.

2 Travaux proches

De nombreuses méthodes ont été proposées pour extraire efficacement des séquences intéressantes à partir de grandes bases de données séquentielles. La plupart s'intéressent à des événements instantanés, représentés par un point sur l'axe des temps, liés par des relations temporelles symboliques basées sur la précédence simple associées à l'utilisation de contraintes globales (par exemple, Srikant et Agrawal (1996); Pei et al. (2001); Zaki (2001); Masseglia et al. (2009)). Pour les événements de durée non nulle, il est nécessaire d'enrichir la représentation des motifs par des relations temporelles, symboliques ou numériques, sur des intervalles.

Concernant l'extraction d'informations temporelles quantitatives, plusieurs algorithmes traitent des séquences d'événements ponctuels datés et en extraient des motifs temporels dont les contraintes temporelles correspondent aux enveloppes de l'ensemble des bornes des intervalles qui sont couverts. Dousson et Duong (1999) ou Cram et al. (2009) ont proposé des méthodes d'extraction de chroniques – graphes de motifs contraints par des relations temporelles quantitatives sur les délais entre événements – à partir de séquences d'événements datés. Giannotti et al. (2006) ou Yoshida et al. (2000) extraient des delta-patterns, règles de la forme $a \xrightarrow{[t_l,t_u]} b$, signifiant que la durée entre a et b est comprise entre t_l et t_u. La caractérisation des intervalles par des enveloppes bénéficie d'une propriété de monotonie dans un treillis de motifs. Cette propriété est efficacement utilisée dans le processus de fouille pour élaguer certains motifs au plus tôt. QFIMiner de Washio et al. (2007) peut être classé dans la même catégorie bien qu'il ne traite pas des séquences mais des ensembles d'items possédant des attributs quantitatifs sous forme d'intervalles. Les motifs extraits sont des ensembles d'items auxquels sont associés des contraintes quantitatives extraites à partir d'une catégorisation des valeurs d'un attribut quantitatif pour les ensembles d'items supportant le motif. Cette classification, combiné avec PrefixSpan, est utilisé dans l'algorithme QTPSpan de Nakagaito et al. (2009).

Néanmoins, la sémantique des motifs induites par l'utilisation de l'enveloppe de séquences n'est pas suffisamment fine pour extraire des motifs comportant les mêmes types d'événements mais associés à des intervalles temporels différents. QTempIntMiner, de Guyet et Quiniou (2008), résout ce problème en proposant une adaption de l'algorithme GSP qui utilise un algorithme de classification de séquences temporelles pour extraire des intervalles temporels typiques. Mais l'utilisation de l'algorithme EM pour cette classification le rend peu efficace.

3 Notations

Dans cette section, nous introduisons les notations et les définitions utiles à la formalisation du problème d'extraction de motifs temporels que nous nous proposons de résoudre.

3.1 Séquence temporelle

Définition 1 (Séquence temporelle)**.** *Une séquence temporelle $\mathcal{S}$ est un ensemble ordonné d'événements, où un événement $\mathcal{A} = (A, [l, u])$, est composé d'un symbole A associé à un intervalle non-vide $[l, u]$, où $l, u \in \mathbb{R}, l < u$, sont des dates.*

$$\mathcal{S} = \{(s_i, [l_i, u_i])\}_{i\in\mathbb{N}_n}\ ,\ tq\ \forall i, j \in \mathbb{N},\ l_i < l_j \vee (l_i = l_j \wedge (s_i < s_j \vee (s_i = s_j \wedge u_i < u_j)))$$

$|\mathcal{S}| = n$, le nombre d'événements, est appelé la taille de la séquence $\mathcal{S}$. On note $\oplus$ l'opération de concaténation de séquences temporelles.

Le début de l'intervalle d'un événement, l_i, donne la date d'occurrence du symbole s_i dans la séquence temporelle. Contrairement à une représentation qui utiliserait des durées entre événements, la définition des valeurs l_i et u_i requiert un instant de référence temporelle pour chaque séquence temporelle.

Définition 2 (Signature symbolique)**.** *La signature symbolique d'une séquence temporelle $\mathcal{S} = \{(s_i, [l_i, u_i])\}_{i\in\mathbb{N}_n}$, notée $\underline{\mathcal{S}}$, est la séquence de ses symboles : $\underline{\mathcal{S}} = \{s_i\}_{i\in\mathbb{N}_n}$. L'ordre de la séquence symbolique conserve l'ordre de la séquence d'intervalles (par dates de début puis par ordre lexicographique).*

Définition 3 (Intervalle multi-dimensionnel)**.** *Un intervalle multi-dimensionnel de dimension n est un n-uplet d'intervalles $I = ([l_i, u_i])_{i\in\mathbb{N}_n}$.*

Définition 4 (Projection sur une signature symbolique π)**.** *La projection d'une séquence temporelle $\mathcal{S} = \{(s_i, [l_i, u_i])\}_{i\in\mathbb{N}_n}$ sur une signature symbolique $\underline{\mathcal{M}} = \{m_i\}_{i\in\mathbb{N}_k}$ de taille k est un intervalle multi-dimensionnel de dimension k.*

Si $\underline{\mathcal{M}}$ n'est pas une sous-séquence de $\underline{\mathcal{S}}$, alors $\pi_{\underline{\mathcal{M}}}(S) = \emptyset$,

Sinon $\pi_{\underline{\mathcal{M}}}(S) = \left([l_{f(i)}, u_{f(i)}]\right)_{i\in\mathbb{N}_k}$, avec $\forall i \in \mathbb{N}_k, s_{f(i)} = m_i$ où $f : \mathbb{N}_n \mapsto \mathbb{N}_k$ est injective, croissante (l'appariement respecte l'ordre des événements dans le motif et dans la séquence).

On note $\pi(\mathcal{S})$ la projection de la séquence entière $\pi_{\underline{\mathcal{S}}}(\mathcal{S}) = ([l_i, u_i])_{i\in\mathbb{N}_n}$.

Exemple 1. *Soit la séquence $\mathcal{E} = \{(A, [1, 2]), (C, [1.5, 4]), (B, [3, 4]), (C, [5, 6])\}$, illustrée par la Figure 1, on a alors $\underline{\mathcal{E}} = \{A, C, B, C\}$, $\pi_{\{A,B\}}(\mathcal{E}) = ([1, 2], [3, 4])$, et $\pi(\mathcal{E}) = ([1, 2], [1.5, 4], [3, 4], [5, 6])$.*

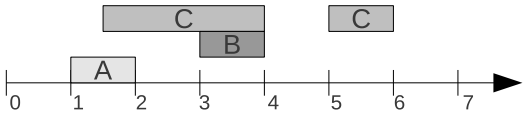

FIG. 1 – *Illustration de la séquence d'intervalles $\mathcal{E}$.*

Définition 5 (Dissimilarité entre séquences temporelles)**.** *Soient $\mathcal{S}^1$ et $\mathcal{S}^2$ deux séquences temporelles, alors la dissimilarité entre $\mathcal{S}^1$ et $\mathcal{S}^2$, notée $d(\mathcal{S}^1, \mathcal{S}^2)$, est définie par :*

$$d(\mathcal{S}^1, \mathcal{S}^2) = \begin{cases} \infty \text{ si } \underline{\mathcal{S}^1} \neq \underline{\mathcal{S}^2}, & \text{(signatures symboliques différentes)} \\ \delta\left(\pi(\mathcal{S}^1), \pi(\mathcal{S}^2)\right) \end{cases}$$

*où δ est une mesure de distance entre intervalles multi-dimensionnels (*cf. *Section 4.4).*

3.2 Extraction de motifs temporels fréquents

Définition 6 (Motif temporel)**.** *Un motif temporel est une séquence temporelle représentative d'un ensemble de séquences temporelles (instances du motifs).*

Tout comme Giannotti et al. (2006), notre définition d'un motif s'appuie sur la notion de représentativité d'une sous-séquence. Les motifs temporels sont construits pour représenter statistiquement un ensemble de séquences temporelles, mais on n'exprime pas formellement la notion de représentativité. Ceci est à distinguer de motifs comme les chroniques ou les delta-patterns dont l'information temporelle est construite par l'enveloppe de l'ensemble des séquences d'événements ponctuels qu'ils recouvrent.

Définition 7 (ϵ-couverture d'un motif temporel)**.** *Un motif temporel $\mathcal{P} = \{(p_i, [l_i^p, u_i^p])\}_{i\in\mathbb{N}_n}$ ϵ-couvre une séquence $\mathcal{S} = \{(s_i, [l_i^s, u_i^s])\}_{i\in\mathbb{N}_m}$, noté $\mathcal{S} \prec_\epsilon \mathcal{P}$, si et seulement si il existe une projection de $\mathcal{S}$ sur $\underline{\mathcal{P}}$ telle que la dissimilarité avec $\mathcal{P}$ soit inférieure à ϵ,* c.-à-d. *:*

$$\exists J \subset \mathbb{N}_m, \quad d\left(\mathcal{P}, \left\{\left(s_j, \left[l_j^s, u_j^s\right]\right)\right\}_{j\in J}\right) < \epsilon$$

Exemple 2. *Considérons trois motifs $p_1 = \{(C, [1, 2]), (A, [2, 4])\}$, $p_2 = \{(A, [1, 2]), (C, [2, 4])\}$ et $p_3 = \{(A, [1, 2]), (C, [4, 5])\}$). Choisissons $\epsilon = 1$ et la distance CityBlock (somme des valeurs absolues des différences entre bornes correspondantes des intervalles,* cf. *Section 4.4) pour δ. p_1 n'ϵ-couvre pas $\mathcal{E}$, la séquence de l'Exemple 1, car la sous-séquence $\{C, A\}$ n'existe pas dans $\mathcal{E}$. p_2 ϵ-couvre $\mathcal{E}$ car δ ($([1, 2], [2, 3.8]), ([1, 2], [1.5, 4])$) $= 0.7 < \epsilon$. p_3 n'ϵ-couvre pas $\mathcal{E}$ car chacune des occurences de p_3 dans $\mathcal{E}$ a une dissimilarité supérieure à ϵ : δ ($([1, 2], [4, 5]),\ ([1, 2], [1.5, 4])$) $= 3.5$ et δ ($([1, 2], [4, 5]),\ ([1, 2], [5, 6])$) $= 2$.*

Définition 8 (Extraction de motifs temporels fréquents)**.** *Étant donnés un seuil de fréquence $f_{min} \in [0, 1]$ et une dissimilarité maximale $\epsilon \in \mathbb{R}^+$, la tâche d'extraction de motifs temporels dans une base de séquences temporelles $\mathcal{D}$ consiste à extraire des motifs temporels tels que la fréquence des séquences qu'ils ϵ-couvrent soit supérieure à f_{min}.*

Un motif temporel est le représentant d'un ensemble de séquences temporelles qu'il ϵ-couvre. Le paramètre ϵ précise la limite de proximité temporelle acceptable entre une séquence et son représentant. Il s'agit d'un paramètre supplémentaire par rapport au problème classique d'extraction de motifs séquentiels fréquents.

4 Présentation des algorithmes

Dans cette partie nous présentons deux algorithmes d'extraction de motifs séquentiels à base d'intervalles temporels : QTIAPriori et QTIPrefixSpan. Tout d'abord, nous introduisons une structure de données utilisée dans ces algorithmes.

4.1 Structure de données

Dans les algorithmes de recherche d'itemsets fréquents ou de séquences fréquentes, le calcul de la fréquence d'un itemset dans la base de séquences est très coûteux en temps de calcul. Pour éviter d'avoir à parcourir systématiquement l'ensemble de la base de séquences,

Algorithme 1 : QTIApriori

input : $\mathcal{D}$: base de données de séquences temporelles, f_{min} : seuil de fréquence, ϵ : seuil de similarité entre motifs, $\delta(.,.)$: mesure de dissimilarité entre intervalles
output : FP : ensemble des motifs temporels fréquents

1 Construction de S_1 *// Symboles fréquents dans* $\mathcal{D}$
2 $k = 1$; $C_1 = S_1$
3 **while** $C_k \neq \emptyset$ **do**
4 $L_k = \emptyset$
5 **foreach** $q \in C_k$ **do**
6 **if** $|\mathcal{L}(q)| > f_{min} * |\mathcal{D}|$ **then** $L_k = L_k \cup q$
7 $FP = FP \cup L_k$
8 **if** $L_k = \emptyset$ **then return** FP
9 *// Génération de nouveaux candidats*
10 $C_{k+1} = \emptyset$
11 **foreach** $p \in L_k$ **do**
12 **foreach** $e \in S_1$ **do**
13 $np = p \oplus e$ *// Extension du motif p*
14 $\mathcal{L}(np) = \mathcal{L}(p)$ *// Attribution a priori d'une liste d'instance*
15 **if** $\forall j \in 1..|np|,\ \exists m \in L_k,\ d(m, np(1..j-1, j+1..|np|)) \leq \epsilon$ **then**
16 $\{Mr\}$ = TemporalPatternConstruction(np) *// Construction des contraintes temporelles et raffinement de* $\mathcal{L}(np)$
17 $C_{k+1} = C_{k+1} \cup Mr$
18 **if** $C_{k+1} = \emptyset$ **then return** FP
19 $k = k + 1$

Srikant et Agrawal (1996) proposent une structure de données associant un motif à l'ensemble des séquences qui supportent un itemset. Nous adaptons cette structure de données pour le comptage des séquences qui supportent un motif séquentiel m.

Pour chaque motif m, $\mathcal{L}(m) = \{stid_i\}$ regroupe l'ensemble de ses instances représentatives. Chaque élément de $\mathcal{L}(m)$ est un triplet $stid = \langle tid, pos, is\rangle$ où tid est l'identifiant d'une séquence temporelle de $\mathcal{D}$, is est une instance représentative du motif m, *c.-à-d.* une sous-séquence de la séquence tid ϵ-couverte par m ($is \prec_\epsilon m$), et pos est la position du dernier symbole de is dans la séquence tid.

$\mathcal{L}(m)$ représente la base projetée sur m, au sens de PrefixSpan (2), *c.-à-d.* l'ensemble des suffixes (localisés par pos) de chaque séquence tid qui a une instance de m pour préfixe.

Exemple 3. *Avec* $\epsilon = 4$*, le motif temporel* p_3 *(*cf. *Exemple 2) ϵ-couvre deux sous-séquences de* $\mathcal{E}$ *(*cf. *Exemple 1, en attribuant* $tid = 1$ *à* $\mathcal{E}$ *dans* $\mathcal{D}$*) :*

$$\mathcal{L}(p_3) = \{\langle 1,\ 2,\ \{(A,[1,2]),(C,[1.5,4])\}\rangle,\ \langle 1,\ 4,\ \{(A,[1,2]),(C,[5,6])\}\rangle\}.$$

4.2 QTIAPriori

QTIAPriori, décrit dans l'algorithme 1, est un algorithme de recherche de motifs temporels fréquents fonctionnant par niveaux à la manière de GSP (Srikant et Agrawal (1996)). Dans une étape préliminaire, C_1, l'ensemble des symboles fréquents dans $\mathcal{D}$ est construit (ligne 1). Ensuite, la boucle principale effectue itérativement les deux étapes classiques de l'algorithme

Algorithme 2 : QTIPrefixSpan

input : $\mathcal{D}$: base de données de séquences temporelles, f_{min} : seuil de fréquence, ϵ : seuil de similarité entre motifs, $\delta(.,.)$: mesure de dissimilarité entre intervalles, p : motif temporel
output : FP : ensemble des motifs temporels fréquents

```
1  Construction de S_1 // Symboles fréquents dans D
2  foreach e ∈ S_1 do
3      //Extension du motif temporel p
4      np = p ⊕ e
5      L(np) = ∅
6      //Projection sur la base de séquences
7      foreach stid ∈ L(p) do
8          s = D(stid.tid)
9          foreach i tq i < stid.pos + 1 ∧ s_i = e do
10             L(np) = L(np) ∪ ⟨stid.tid, i, stid.is ⊕ s_i⟩
11     //Construction des contraintes temporelles et récursion
12     {Mr} = TemporalPatternConstruction(np)
13     foreach q ∈ {Mr} do
14         if q ∉ already_explored then
15             already_explored = already_explored ∪ q
16             if |L(q)| > f_min * |D| then
17                 ExtentedPatterns = QTIPrefixSpan( D, f_min, ϵ, δ(.,.), q )
18                 FP = FP ∪ ExtentedPatterns
```

GSP : l'élagage des motifs candidats non-fréquents (lignes 5-6) puis la génération de nouveaux motifs candidats de taille $k+1$ à partir des motifs fréquents de taille k (lignes 10-17). Le cycle s'arrête lorsqu'il n'y a plus de motifs temporels fréquents à étendre.

L'élagage de la première étape ne nécessite pas de comptage explicite des instances d'un motif puisque celles-ci sont représentées au moyen de la structure de données $\mathcal{L}(q)$. Si $\mathcal{L}(q)$ contient suffisamment d'instances (ligne 5), q est ajouté aux motifs temporels fréquents L_k.

Dans la seconde étape (lignes 10-17), les motifs candidats sont composés d'un motif appartenant à L_k suivi d'un événement constitué d'un symbole e issu de S_1. Initialement, e n'est associé à aucun intervalle temporel. À la ligne 14, on attribue provisoirement à np l'ensemble des instances associées à p. Comme p est un sous-motif de np, on sait *a priori* que la liste des instances ϵ-couvertes par np est contenue dans la liste des instances ϵ-couvertes par p.

L'étape de construction des contraintes temporelles étant coûteuse en temps, il est plus efficace de pré-sélectionner les candidats ayant des chances d'être fréquents. Pour cela, on utilise la contrainte APriori exprimée ligne 15 : tous les sous-motifs de np de taille k doivent être similaires à au moins un des motifs m de L_k. Dans la mesure où e n'est associé à aucun intervalle temporel, le calcul de la similarité ignore la dimension temporelle de e. Ensuite, la fonction TemporalPatternConstruction (ligne 16) construit une liste de motifs temporels $\{Mr\}$ ayant la même signature symbolique que np dont les caractéristiques temporelles sont obtenues par classification des intervalles multidimensionnels (*cf.* Section 4.4). Cette fonction se charge également de mettre à jour les structures de données pour chaque nouveau motif candidat.

Algorithme 3 : TemporalPatternConstruction

input : p : motif temporel avec $\mathcal{L}(p)$ qui contient la liste des instances du sous-motif de p
output : $\{Mr\}$: ensemble des motifs temporels de signature symbolique $\underline{p}$ temporellement représentatifs des instances dans $\mathcal{L}(p)$

1 *//Construction de l'ensemble des intervalles temporels à catégoriser*
2 $Ex = \emptyset$
3 **foreach** $stid \in \mathcal{L}(p)$ **do**
4 $\quad Ex = Ex \cup \pi(stid.is)$
5 $\mathcal{C} = categoriser(Ex)$
6 *//Construction des motifs temporels*
7 $\{Mr\} = \emptyset$
8 **foreach** $C \in \mathcal{C}$ **do**
9 $\quad q = \left\{(\underline{p}_i, c_i)\right\}_i$
10 $\quad \mathcal{L}(q) = \{stid \in \mathcal{L}(p),\ d(stid.is, q) < \epsilon\}$
11 $\quad \{Mr\} = \{Mr\} \cup q$

4.3 QTIPrefixSpan

QTIPrefixSpan (algorithme 2), est un algorithme récursif de type profondeur d'abord basé sur PrefixSpan (Pei et al. (2001)). Il explore les extensions d'un motif temporel p, fourni en paramètre, en effectuant des projections successives de la base de séquences afin de réduire la complexité du calcul du support et de la catégorisation temporelles des motifs étendus.

Dans une première phase (lignes 3-5), l'algorithme étend le préfixe p avec l'un des symboles e de S_1 (S_1 est construit lors d'une étape préliminaire à la récursion). L'algorithme effectue ensuite la projection de la base de séquences sur le motif $np = p \oplus e$ (lignes 6-10). Comme la liste $\mathcal{L}(np)$ représente la base projetée (*cf.* Section 4.1), cette étape consiste à construire la liste de $stid$ associée à np. Ici aussi, la dimension temporelle de e est ignorée.

Dans la seconde phase (lignes 11-18), de même que pour QTIAPriori, la fonction TemporalPatternConstruction construit des motifs temporels de même signature que np. Les motifs temporels non-fréquents sont ensuite élagués (ligne 16). Si le motif q est fréquent, alors il est ajouté aux motifs explorés. Enfin, QTIPrefixSpan est appelé récursivement pour étendre q.

4.4 Catégorisation d'intervalles multi-dimensionnels

L'algorithme TemporalPatternConstruction(p) (*cf.* Algorithme 3) est utilisé dans les algorithmes QTIAPriori (ligne 16) et QTIPrefixSpan (ligne 12). Il construit des motifs temporels représentatifs d'un ensemble de sous-séquences temporelles ayant la même signature symbolique. La méthode consiste 1) à catégoriser l'ensemble des intervalles multidimensionnels obtenus par la projection des $stid.is$ contenus dans $\mathcal{L}(p)$, 2) à construire, pour chacune des catégories, un nouveau motif q ayant la même signature symbolique que p et avec les contraintes temporelles caractéristiques des intervalles multidimensionnels de la catégorie, 3) à construire $\mathcal{L}(q)$ à partir de $\mathcal{L}(p)$.

Pour la catégorisation réalisée à la ligne 5, nous utilisons un algorithme KMeans ou un algorithme Affinity Propagation (AP) de Frey et Dueck (2007), que nous avons adaptés à la catégorisation d'intervalles multi-dimensionnels. Pour ce faire, nous avons sélectionné deux similarités entre les objets à classer, ici des intervalles multi-dimensionnels $I = ([l_k^I, u_k^I])_{k \in \mathbb{N}_n}$

et $J = ([l_k^J, u_k^J])_{k \in \mathbb{N}_n}$ de dimension n : la distance de Haussdorf, notée d_H, et la distance City-Block, notée d_{CB}. D'autres mesures de distance entre intervalles multi-dimensionnels peuvent être trouvées dans Irpino et Verde (2008).

$$d_H(I,J) = \sum_{k=1}^{n} \max\left(|l_k^I - l_k^J|, |u_k^I - u_k^J|\right), \quad d_{CB}(I,J) = \sum_{k=1}^{n} |l_k^I - l_k^J| + |u_k^I - u_k^J|$$

À la ligne 10, la liste des instances du motif temporel q est mise à jour. Comme avec les bases projetées de PrefixSpan, on évite de parcourir l'ensemble de la base de séquences en ne cherchant les instances du motif temporel q qu'à partir des instances d'un sous-motif de q. En entrée de l'algorithme, on pose comme précondition que $\mathcal{L}(p)$ contient les instances du sous motif de p : $p(1..|p|-1)$, on construit donc $\mathcal{L}(q)$ à partir de cette liste de $stid$.

Pour trouver le nombre de classes intrinsèques à la distribution d'intervalles, il faut tester différentes valeurs de k pour les K-Means, ou du paramètre s pour AP, et choisir celle qui convient le mieux aux données. Pour sélectionner le nombre de classes, nous utilisons le critère BIC (Bayesian Information Criterion).

5 Expérimentations

Les algorithmes QTIApriori et QTIPrefixSpan ont été implémentés en Matlab [1]. Dans les expérimentations suivantes, nous comparons les performances en temps de calcul des algorithmes proposés et la qualité des motifs extraits. Les résultats sont comparés avec ceux de QTempIntMiner. Les expérimentations utilisent des données simulées.

5.1 Production de données simulées

Nous avons développé un générateur de séquences temporelles afin de produire des jeux de données paramétrables selon les caractéristiques des motifs à rechercher (en particulier leurs caractéristiques temporelles numériques) et celles de la base de séquences (nombre de séquences, proportion de bruit, ...). Le principe de la génération d'une base de séquences temporelles consiste, d'une part, à construire des séquences à partir d'un ou plusieurs prototypes de motif temporel et, d'autre part, à ajouter des séquences aléatoires (bruit) dans la base de séquences selon une proportion rP comprise entre 0 et 1.

Un prototype de motif temporel est un ensemble de n 5-uplets $\{(E_i, \mu_{b_i}, \sigma_{b_i}, \mu_{d_i}, \sigma_{d_i})\}_{i \in \mathbb{N}_n}$ où E_i est un symbole, μ_{b_i}, μ_{d_i} (resp. $\sigma_{b_i}, \sigma_{d_i}$) sont les moyennes (resp. écarts-types) de la date de début et de la durée de E_i. Un prototype spécifie un motif temporel à découvrir $(\{(E_i, [\mu_{b_i}, \mu_{b_i} + \mu_{d_i}])\}_{i \in \mathbb{N}_n})$. Les instances de ce motif sont générées en utilisant une distribution Gaussienne des dates (resp. durées) centrée sur μ_b (resp. μ_d) avec un écart-type σ_b (resp. σ_d). Plus l'écart-type est important, plus les variations temporelles sont importantes et plus l'extraction du motif original sera difficile. Pour simplifier la présentation des résultats, nous fixons un unique paramètre $tN \in \mathbb{R}^+$ pour quantifier tous les écarts-types. Ce paramètre quantifie le *bruit temporel* d'un jeu de données.

1. Les expérimentations ont été menées sur un ordinateur ayant un processeur Xeon (un seul cœur utilisé), avec 8Go de RAM. Les sources des algorithmes et le simulateur de données peuvent être téléchargés sur le site : *www.irisa.fr/dream/QTempIntMiner/*

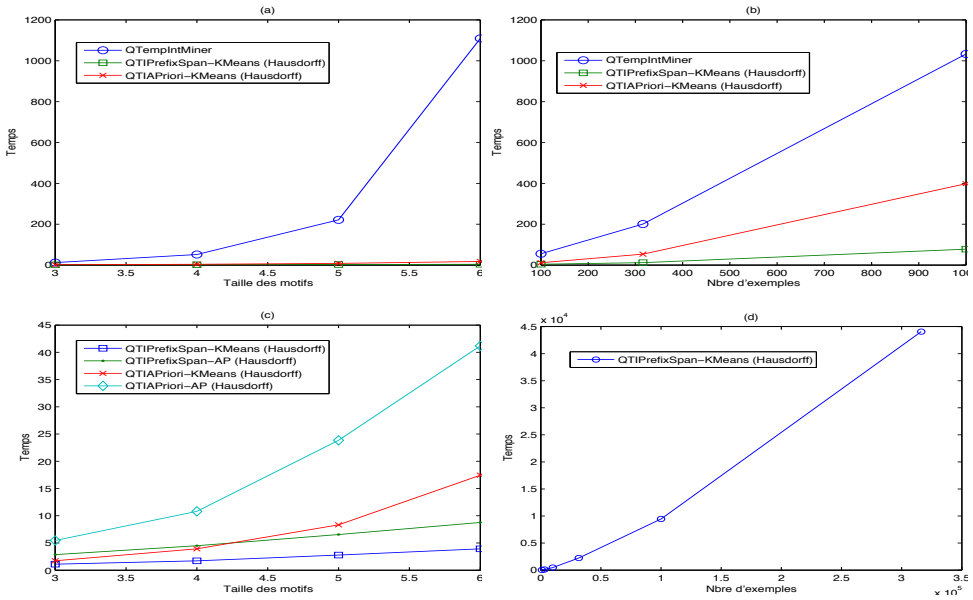

FIG. 2 – *Temps de calcul moyen (en secondes) en fonction de la taille des motifs temporels recherchés (a et c) et en fonction du nombre de séquences (b et d).*

5.2 Résultats

Toutes les courbes présentées dans la suite ont été obtenues en moyennant les résultats sur plusieurs jeux de données différents (entre 5 et 10), générés avec les mêmes paramètres. Sans indication contraire, les paramètres ont les valeurs par défaut suivantes : $|\mathcal{D}| = 100$, $rP = 0.4$, $tN = 0.2$, $f_{min} = 0.1$ et $\epsilon = \infty$. La distance de Hausdorff est utilisé et le motif temporel à extraire est le suivant :

$$\{(A, [2, 3]), (B, [5, 5.5]), (C, [8, 10]), (B, [12, 12.5])\}.$$

Les Figures 2-(a) et 2-(b) comparent les temps de calcul des algorithmes QTempIntMiner, QTIAPriori et QTIPrefixSpan en fonction, d'une part, de la taille de la base de séquences et, d'autre part, de la taille des motifs recherchés. Dans les deux cas, on constate que les temps de calcul des trois algorithmes sont exponentiels. De plus, QTIAPriori et QTIPrefixSpan sont significativement plus rapides que QTempIntMiner.

Plus précisément (*cf.* Figure 2-(c)), QTIPrefixSpan est deux fois plus rapide que QTIA-Priori, qu'il soit associé à KMeans ou à AP. La classification par KMeans mène à de meilleures performances que la classification par AP. QTIAPriori-KMeans est même plus rapide que QTIPrefixSpan-AP pour des motifs de taille supérieure à 4. La Figure 2-(d) illustre la capacité de QTIPrefixSpan-KMeans à passer à l'échelle. La croissance du temps de calcul en fonction du nombre de séquences est exponentielle mais de coefficient 1.26, soit relativement proche de 1. Le temps moyen de traitement d'une base de données de 100000 séquences, soit environ 600000 symboles, est d'environ 15 min pour extraire des motifs temporels de taille 4.

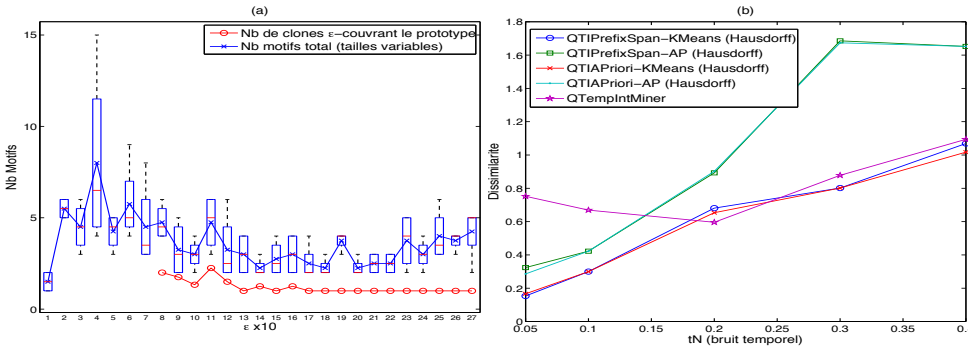

FIG. 3 – *À gauche : nombre moyen de motifs extraits en fonction de ϵ pour l'algorithme QTIPrefixSpan-KMeans. À droite : dissimilarité moyenne entre le meilleur motif extrait et le prototype en fonction du bruit temporel.*

La qualité des motifs extraits par QTIAPriori et QTIPrefixSpan peut être évaluée selon différents critères : i) la présence ou l'absence d'un motif ϵ-couvrant le prototype parmi les motifs extraits ; ii) le nombre de motifs extraits n'ϵ-couvrant pas le prototype ; iii) la dissimilarité entre les intervalles temporels du prototype et ceux des motifs extraits, *c.-à-d.* la précision des intervalles temporels associés aux motifs extraits. iv) le nombre de clones du prototype (les motifs de même taille que le prototype et d'une dissimilarité inférieure à ϵ) figurant parmi les motifs extraits. Ce dernier critère est lié à la facilité de lecture des résultats : plus le nombre de clones est bas, plus le traitement des résultats sera facile pour l'utilisateur.

Les facteurs principaux affectant ces critères sont le niveau de bruit temporel tN, le seuil de dissimilarité ϵ et le seuil de fréquence minimale f_{min}. Ces paramètres sont liés : si les données sont très bruités, alors un ϵ grand sera nécessaire pour que les instances bruitées d'un motif soient classées dans une même catégorie. Mais si ϵ est trop grand (par rapport à la valeur de tN) alors les classes seront plus larges et l'élagage par utilisation de f_{min} sera moins performant au détriment de l'efficacité de l'algorithme et du nombre de motifs extraits.

Pour ϵ suffisamment grand et f_{min} suffisamment petit, les trois algorithmes extraient le motif prototype des jeux de séquences. De plus, si on ajoute un second prototype avec la même signature symbolique que le premier mais des intervalles temporels différents, alors les deux motifs sont correctement extraits (un seul extrait par les algorithmes GSP ou PrefixSpan).

La Figure 3 illustre la qualité des motifs extraits par QTIAPriori, QTIPrefixSpan et QTempIntMiner à partir de données simulées ne contenant qu'un seul motif fréquent au milieu de séquences aléatoires. La Figure 3-(a) illustre, pour l'algorithme QTIPrefixSpan, la variation du nombre de motifs de FP (ligne 18 de l'algorithme 2) et le nombre de clones du prototype, en fonction de ϵ. Pour ϵ petit ($\epsilon < .1$), peu de motifs de la taille du prototype sont construits : la similarité attendue entre les instances d'un motif impose une contrainte trop forte par rapport au bruit des données. Ensuite, le nombre de motifs construits grandit fortement, mais la classification temporelle produit un nombre de classes important, si bien que le nombre d'instances d'un motif associé à une classe reste inférieur à f_{min}. À partir de $\epsilon = .8$ plusieurs motifs fréquents sont extraits, puis leur nombre se stabilise à 1 à partir de $\epsilon = 1.7$. ϵ est alors suffisamment grand pour considérer que les dissimilarités entre intervalles temporels sont acceptables.

On constate sur la Figure 3-(b) que même avec du bruit, le meilleur motif temporel extrait par les trois algorithmes est toujours très proche du motif recherché : une distance de Hausdorff de 1 est faible pour un motif temporel ayant des valeurs d'intervalle allant jusqu'à 12 unités de temps. Pour QTIAPriori et QTIPrefixSpan, il se dégage logiquement que plus le bruit temporel augmente, plus la précision diminue. On constate également que l'algorithme de catégorisation influence le plus la résistance au bruit, et ce indépendamment de la méthode d'exploration des motifs. KMeans résiste mieux au bruit temporel que AP. La dissimilarité moyenne obtenue avec KMeans est inférieure à celle obtenue avec AP. QTempIntMiner se montre aussi précis, voire plus précis, que les deux autres algorithmes pour les motifs de grande taille (catégorisation par EM de qualité supérieure à celle par KMeans ou AP).

6 Conclusion

Nous avons présenté le problème de l'extraction de motifs temporels à partir de séquences d'événements qualifiés par une date et une durée *c.-à-d.* associés à des intervalles temporels numériques. De tels motifs sont particulièrement expressifs : ils caractérisent et discriminent les séquences d'une base de séquences en entrée non seulement sur la succession des événements, mais également sur leurs positions relatives dans le temps et sur leurs durées. Nous avons proposé deux algorithmes, QTIAPriori et QTIPrefixSpan, pour extraire des motifs temporels contenant des informations temporelles quantitatives. Ils associent une étape de catégorisation d'intervalles multidimensionnels aux deux algorithmes classiques d'extraction de motifs séquentiels. La comparaison avec QTempIntMiner montre que nos algorithmes sont plus efficaces en temps de calcul et extraient des motifs temporels plus précis. Des trois algorithmes, QTIPrefixSpan est le plus efficace en temps de calcul sur nos jeux de données simulées.

Les améliorations apportées à la fouille de séquences d'intervalles permettent de traiter efficacement un grand volume de données. Des applications pratiques, par exemple, pour la caractérisation de pathologies cardiaques à partir d'électrocardiogrammes doivent maintenant mettrent en évidence l'intérêt pratique de tels motifs temporels. La catégorisation demeure la source de complexité majeure des algorithmes proposés. Nous souhaitons étudier différentes pistes pour en améliorer l'efficacité, comme le choix heuristique a priori du nombre de classes ou l'introduction d'une certaine incrémentalité dans les méthodes de classification.

Références

Cram, D., A. Cordier, et A. Mille (2009). An interactive algorithm for the complete discovery of chronicles. Technical Report RR-LIRIS-2009-011, LIRIS UMR 5205 CNRS.

Dousson, C. et T. Duong (1999). Discovering Chronicles with Numerical Time Constraints from Alarm Logs for Monitoring Dynamic Systems. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pp. 620–626.

Frey, B. et D. Dueck (2007). Clustering by passing messages between data points. *Science 315*, 972–976.

Giannotti, F., M. Nanni, D. Pedreschi, et F. Pinelli (2006). Mining sequences with temporal annotations. In *Proceedings of the Symposium on Applied Computing*, pp. 593–597.

Guyet, T. et R. Quiniou (2008). Mining temporal patterns with quantitative intervals. In *Proceedings of the 4th International Workshop on Mining Complex Data*, pp. 10.

Irpino, A. et R. Verde (2008). Dynamic clustering of interval data using a wasserstein-based distance. *Pattern Recognition Letters 29*, 1648–1658.

Le Ber, F., M. Benoît, C. Schott, J.-F. Mari, et C. Mignolet (2006). Studying crop sequences with CarrotAge, a HMM-based data mining software. *Ecological Modelling 191*, 170–185.

Masseglia, F., P. Poncelet, et M. Teisseire (2009). Efficient mining of sequential patterns with time constraints : reducing the combinations. *Expert Systems with Applications 36*(2), 2677–2690.

Nakagaito, F., T. Ozaki, et T. Ohkawa (2009). Discovery of quantitative sequential patterns from event sequences. In *Proceedings of the International Conference on Data Mining Workshops*, pp. 31–36.

Pei, J., J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, et M.-C. Hsu (2001). PrefixSpan mining sequential patterns efficiently by prefix projected pattern growth. In *Proceedings of International Conference on Data Engineering*, pp. 215–226.

Quiniou, R., G. Carrault, M.-O. Cordier, et F. Wang (2003). Temporal abstraction and inductive logic programming for arrythmia recognition from electrocardiograms. *Artificial Intelligence in Medicine 28*, 231–263.

Srikant, R. et R. Agrawal (1996). Mining sequential patterns : Generalizations and performance improvements. In *Proceedings of the Fifth International Conference on Extending Database Technology*, pp. 3–17.

Washio, T., K. Nakanishi, et H. Motoda (2007). A classification method based on subspace clustering and association rules. *New Generation Computing 25*, 235–245.

Yoshida, M., T. Iizuka, H. Shiohara, et M. Ishiguro (2000). Mining sequential patterns including time intervals. In *Proceedings of the conference on Data Mining and Knowledge Discovery : theory, tools, and technology*, pp. 213–220.

Zaki, M. J. (2001). Spade : An efficient algorithm for mining frequent sequences. *Machine Learning 42*(1/2), 31–60.

Summary

Most of the sequential patterns extraction methods proposed so far deal with patterns composed of events linked by temporal relationships based on simple precedence between instants. In many real situations, some quantitative information about event duration or the inter-event delay is needed in order to discriminate phenomena. We propose two algorithms, QTIAPriori and QTIPrefixSpan, for extracting temporal patterns composed of events to which are associated temporal intervals describing their position in time and their duration. Each of them, adds to algorithms GSP and PrefixSpan a multi-dimensional interval categorisation step for extracting representative temporal intervals. Experiments on simulated data show that both algorithms are efficient for extracting precise patterns even in noisy contexts and improve the performance of an older algorithm using a categorisation method based on Expectation-Maximization.